Predicting Protein Function and Protein-Ligand Interaction with the 3D Neighborhood Kernel

Leander Schietgat, Thomas Fannes, and Jan Ramon

Department of Computer Science, KU Leuven
Celestijnenlaan 200A, 3001 Leuven, Belgium
{firstname.lastname}@cs.kuleuven.be

Abstract. Kernels for structured data have gained a lot of attention in a world with an ever increasing amount of complex data, generated from domains such as biology, chemistry, or engineering. However, while many applications involve spatial aspects, up to now only few kernel methods have been designed to take 3D information into account. We introduce a novel kernel called the 3D Neighborhood Kernel. As a first step, we focus on 3D structures of proteins and ligands, in which the atoms are represented as points in 3D space. By comparing the Euclidean distances between selected sets of atoms, the kernel can select spatial features that are important for determining functions of proteins or interactions with other molecules. We evaluate the kernel on a number of benchmark datasets and show that it obtains a competitive performance w.r.t. the state-of-the-art methods. While we apply this kernel to proteins and ligands, it is applicable to any kind of 3D data where objects follow a common schema, such as RNA, cars, or standardized equipment.

1 Introduction

Over the past years, kernel functions for structured data have gained a lot of attention and were successfully applied to many real-world problems. Chemoinformatics is an area which is of particular interest. Since molecules are naturally represented by graphs, graph kernels have proven very suitable for this kind of problems and they have obtained excellent results [21, 5, 6]. However, until now, attention has mostly focused on the so-called small (mostly drug) molecules. In this context, even NP-hard problems can usually still be solved efficiently in practice. The ability to tackle proteins, which are two orders of magnitude larger, is a far bigger challenge.

Proteins are macromolecules that play a crucial role in a wide range of biological processes. They are responsible for, e.g., signaling responses between or within cells, the formation of structural elements, or catalyzing chemical reactions. In order to obtain more insights into these processes, many of them have been modeled by machine learning and data mining methods, with applications as predicting the function of proteins [24], the ligands they bind to [15, 6], or drug resistance in HIV [8].

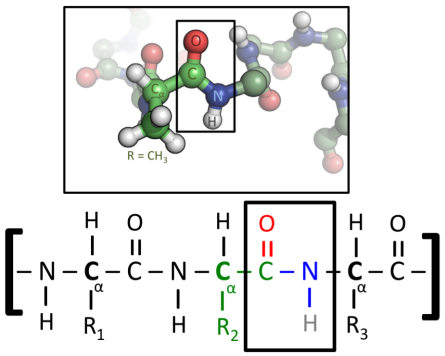


Fig. 1. Chemical structure of the peptide bond (bottom) and the three-dimensional structure of a peptide bond between an Alanine and an adjacent amino acid (top) (from: Wikipedia/Protein). The black box indicates a single peptide bond. We call all the atoms belonging to the side chains or residues (abbreviated as R_1, R_2, R_3 in the figure) side chain atoms. We call all other atoms backbone atoms.

A protein consists of a polypeptide chain of amino acids linked with peptide bonds (Fig. 1). The linked series of carbon, nitrogen and oxygen atoms is known as the protein backbone and the variable parts of the amino acids are the residues or side chains. Proteins can be represented as graphs using different approaches [22]. In some approaches, atoms are represented as vertices and bonds as edges whereas others use nodes to represent amino acids and edges to represent the strength of interaction between the side chains of two amino acids. The structural similarities can be measured with graph mining tools such as kernels [21] or distance measures [25]. Analyzing structural similarity is motivated by the fact that proteins having similar structures are more likely to exhibit common biochemical properties [2].

However, existing graph kernels have multiple limitations. First, they perform poorly on large labeled graphs [26]. Second, they do not take into account 3D information directly. By transforming the protein structures into graphs, information about angles and exact distances is lost. Third, the size of protein graphs have a large impact on their efficiency. In general, it is not clear yet whether learning on 3D structures directly results in accurate models and whether this approach performs better than state-of-the-art graph kernels applied to proteins.

In this paper, we propose a new kernel for 3D data, called the 3D Neighborhood Kernel (3DNK). In contrast to existing kernels, it takes spatial distances into account, focusing on geometry rather than relationships in a graph. As a first step, we focus on two biological applications involving 3D structures of proteins and ligands. The first task involves the classification of proteins into enzymes and Gene Ontology (GO) classes [20], while the second task involves the prediction of binding affinities, representing interaction strength between proteins

and ligands [1]. We will compare our kernel to four state-of-the-art methods: two graph kernels (the Fast Subtree Kernel (FSTK) [26] and the Neighborhood Subgraph Pairwise Distance Kernel (NSPDK) [6]) and two methods that were created specifically to solve the aforementioned biological tasks (the MAMMOTH kernel [20] and RF-Score [1]). Our results show that the 3DNK kernel is competitive with the state-of-the-art methods w.r.t. predictive performance, while it can be applied to a larger variety of tasks.

The remainder of the paper is structured as follows. Section 2 presents the necessary definitions and notation w.r.t. support vector machines and kernel methods. We describe the 3DNK kernel in Section 3. Section 4 performs an experimental evaluation of 3DNK and a comparison with the state-of-the-art methods. Section 5 discusses related work and Section 6 concludes.

2 Preliminaries

Kernel functions Given a set X and a function $K : X \times X \to \mathbb{R}$, we say that K is symmetric if for any x_i and $x_j \in X$ holds that $K(x_i, x_j) = K(x_j, x_i)$, and K is positive-semidefinite if for any $n \geq 1$ and any $x_1, \ldots, x_n \in X$, the matrix K defined by $K(x_i, x_j)$ is positive-semidefinite, that is, $\sum_{ij} c_i c_j K(x_i, x_j) \geq 0$ for all $c_1, \ldots, c_n \in \mathbb{R}$ or equivalently if all its eigenvalues are nonnegative. The function K is called a kernel function and $K(x_i, x_j)$ represents a measure of similarity between x_i and x_j, which can be for example vectors, strings, trees, graphs, or 3D structures.

Support vector machines (SVMs) Let X be a non-empty set of n training examples associated with class labels $\{\mathbf{x}_i, y_i\}_{i=1}^n$, $\mathbf{x}_i \in X = \mathbb{R}^d$, $d \in \mathbb{N}$ the dimension of input space, and $y_i \in \mathbb{R}$ the target value (discrete in the classification case, a numerical value in the regression case). The task is to learn a function $f : X \to y$ to predict the target value of unlabeled examples. An SVM gives the solution to, for example, the binary classification problem by introducing a hyperplane that separates the training data into positive and negative examples. An infinite number of such hyperplanes exists. Let $\mathbf{w} \in \mathbb{R}^d$ be the weight vector that determines the orientation of the hyperplane and $b \in \mathbb{R}$ be the threshold that determines the offset of the plane from the origin. The class of such hyperplanes is then given by $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$ and corresponds to the decision function $f(\mathbf{x}) = sgn(\langle \mathbf{w}, \mathbf{x} \rangle + b)$. The maximum margin classifier identifies the optimal hyperplane that is distinguished by the maximum distance from the nearest training objects in both the classes. The optimal solution is unique and sparse, and it is determined by data points close to the decision boundaries called *support vectors*. In the regression case, a similar method is applied, but the orientation of the hyperplane is determined by the ϵ-sensitive loss, which measures the deviation from the target values y_i [7].

In many real-world situations, the data are not easily separable in the input space. However, linear separation can be achieved if the input data are projected onto some higher dimensional dot product feature space F. Let $\phi : X \to F$ be a non-linear mapping from input space X to feature space F. Surprisingly, the

explicit mapping of data from input space to feature space is not needed. A kernel function, mapping $K(x_i, x_j)$ to $\langle \phi(x_i), \phi(x_j) \rangle$ makes it possible to compute dot products in the feature space without explicitly knowing the mapping ϕ [28, 7].

3 The 3D Neighborhood Kernel

In this section, we will first present a family of kernels that act on 3D point sets (Sect. 3.1). Then, we discuss how we instantiate different kernels from this family in order to solve two specific biological tasks (Sect. 3.2). Finally, we show how we can make the kernel more efficient (Sect. 3.3).

3.1 The 3D Neighborhood Kernel Family

Let $P = \mathbb{R}^3 \times \Sigma$ be the set of all 3D points, embedded in a Euclidian space and labeled over an alphabet Σ. For a point $p \in P$, let $\zeta(p)$ and $\lambda(p)$ represent its 3D coordinates, i.e., a tuple $\langle x_p, y_p, z_p \rangle$, and its label, respectively. Furthermore, let the input space $\mathcal{X} = 2^P$ represent the set of all possible 3D point sets. We will call a point set $X \in \mathcal{X}$ an example.

Let $n \in \mathbb{N}$ be a parameter. Let $\mathcal{F}_\Delta$ be the family of all functions $\Delta : \mathcal{X} \to \mathcal{X}$ for which $\forall X : \Delta(X) \subseteq X$, i.e., for any point set, Δ outputs a subset of that point set. Let $\mathcal{F}_\Phi$ be the family of all functions $\Phi : \mathcal{X} \times P \to P^n$ such that for every $X \in \mathcal{X}$ and $p \in X$, $\Phi(X, p) = \langle p_1, p_2, \ldots, p_n \rangle \in X^n$ is a tuple of n points with $p_i \in X$ for $1 \leq i \leq n$. For a function $\Phi \in \mathcal{F}_\Phi$, we define:

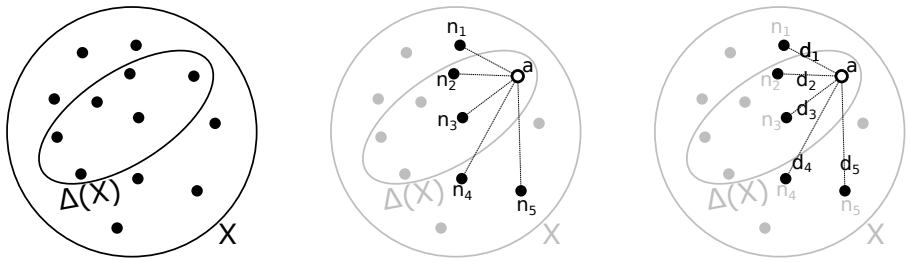
$$d_\Phi(X, p) = \langle \|\zeta(p) - \zeta(p_1)\|, \|\zeta(p) - \zeta(p_2)\|, \ldots, \|\zeta(p) - \zeta(p_n)\| \rangle,$$

where $\Phi(X, p) = \langle p_1, p_2, \ldots, p_n \rangle$. This function generates a tuple of Euclidian distances, where the distances are those from p to the corresponding point decided by $\Phi(X, p)$.

The idea of the 3D Neighborhood Kernel (3DNK) is to compare point sets based on their 3D structure. More specifically, the kernel performs the following steps on its two input point sets: *(i)* for each of both point sets, a subset of points is selected (called the *selected* points) according to a user-specified criterion Δ; *(ii)* for each selected point, its neighborhood is retrieved according to a user-specified neighborhood function Φ; and *(iii)* for each point in the sets of selected points, a feature vector is constructed describing the local spatial conformation of that point in its neighborhood. The distance between two point sets is then calculated by comparing the feature vectors of all pairs of identically labeled, selected points. The construction of a tuple of distances for a point $a \in \Delta X$ is shown in Fig. 2.

Definition 1 (3DNK family). *Let $n \in \mathbb{N}$ be a neighborhood size parameter, $\Delta \in \mathcal{F}_\Delta$ be a selection function, $\Phi \in \mathcal{F}_\Phi$ be a neighborhood function, and $\sigma \in \mathbb{R}^+$ be a parameter for the Gaussian RMS width. The 3DNK family, $K_{\Delta,\Phi} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, is defined as follows:*

$$K_{\Delta,\Phi}(X, Y) = \sum_{a \in \Delta(X)} \sum_{b \in \Delta(Y)} K_G \left(d_\Phi(X, a), d_\Phi(Y, b) \right) \cdot I \left(\lambda(a) = \lambda(b) \right),$$



(a) An example X and its $\Delta(X)$.

(b) Neighborhood for a selected point: $\Phi(X, a) = \langle n_1, n_2, n_3, n_4, n_5 \rangle$.

(c) $d_\Phi(X, a) = \langle d_1, d_2, d_3, d_4, d_5 \rangle$, where d_i is the distance between a and the corresponding neighborhood point n_i.

Fig. 2. Overview of the defined functions Δ, Φ and d_Φ.

where $K_G : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ *is a Gaussian-based distance kernel, i.e.,*

$$K_G(\boldsymbol{v_a}, \boldsymbol{v_b}) = exp\left(\frac{-\|\boldsymbol{v_a} - \boldsymbol{v_b}\|^2}{\sigma^2}\right),$$

and $I(x) = 1$ *if* x *is true, and 0 otherwise.*

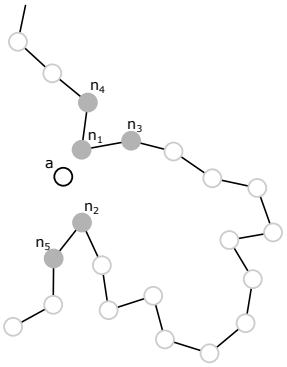
It can be easily verified that instantiations of the 3DNK family will lead to positive-semidefinite kernels.

3.2 Instantiations of the 3DNK Kernel

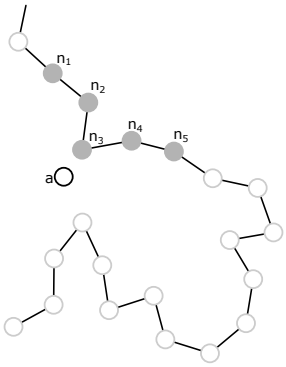
In this section, we discuss how we instantiate the two function parameters of the 3DNK family in order to solve two biological tasks: predicting protein function and protein-ligand interaction. We denote with $\mathcal{C}$ the set of all chemical elements.

Predicting Protein Function In this setting, an example is a protein 3D structure, consisting of atoms. For such a protein X and an atom $a \in X$, we define $\Delta(X)$ as the set of its side chain atoms, and the neighborhood function $\Phi(X, a)$ will only select backbone atoms of X in the neighborhood of a. The motivation for this is that the distances between atoms from the backbone and atoms from the side chains will influence binding pocket geometry, and hence determine protein function. For example, when predicting resistance in HIV proteins, resistant proteins will generally have similar backbones, but the acquired mutations (changing the side chains of the protein) will influence the distances between backbone and side chain atoms.

Let X be a protein, $a \in \Delta(X)$ a side chain atom, and $\Sigma = \mathcal{C}$. We present two approaches for mapping a side chain atom to a feature vector, i.e., two candidates for the function Φ as given in Definition 1, resulting into two kernels:



(a) Nearest neighbor neighborhood selection.



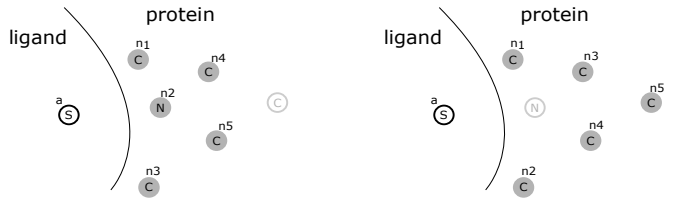
(b) Sequence window neighborhood selection centered around the nearest backbone atom (n_3).

Fig. 3. The backbone atoms n_1 to n_5 for an atom a (neighborhood size 5).

1. **Nearest neighbor** For each side chain atom a, $\Phi_{nn}(X, a)$ maps the atom to the n nearest backbone atoms, ordered in ascending order (Fig. 3(a)). We denote this kernel with 3DNK$_{nn}$.
2. **Sequence window** For each side chain atom a, $\Phi_{sw}(X, a)$ maps the atom to n backbone atoms, by using a window of size n over the backbone, centered around its nearest backbone atom. The tuple of size n consists of the backbone atoms in the window, ordered from left to right (Fig. 3(b)). Note that amino acids towards both ends of the sequence do not have a complete window, and therefore these side chain atoms are not used in the kernel. We denote this kernel with 3DNK$_{sw}$.

Predicting Protein-Ligand Interaction In this setting, an example X is the set of ligand atoms and the protein binding pocket atoms to which the ligand is bound. We define $\Delta(X)$ as the set of ligand atoms, while the neighborhood function Φ will select atoms from the protein binding pocket. The motivation for this is that the distances between the ligand atoms and the atoms from the binding pocket will influence the binding affinity. We again present two approaches:

1. **Ligand type** For each ligand atom a, we construct the neighborhood by using Φ_{nn} to select the nearest protein atoms (Fig. 4(a)). We reuse the notation 3DNK$_{nn}$, as it is very similar to the one of the previous learning task.
2. **Ligand-protein atom type** Contrary to 3DNK$_{nn}$, which does not take into account the atom types at the protein side, here we construct for each



(a) Nearest neighbor neighborhood selection.

(b) Nearest neighbor, atom-type dependent neighborhood selection of atom a and neighbors with label C, $\lambda(a) = (S, C)$.

Fig. 4. The protein atoms n_1 to n_5 for an atom a (neighborhood size 5).

ligand atom a and each atom type t the neighborhood by selecting the nearest protein atoms having an atom type t. Constructing multiple neighborhoods per ligand atom (one for each atom type), can be done by applying the following procedure: *(i)* for each ligand atom $a \in \Delta(X)$ and for each atom type $t \in \Sigma$, add a new point p to $\Delta(X)$ with $\zeta(p) = \zeta(a)$ and $\lambda(p) = (\lambda(a), t) \in \Sigma^2$; *(ii)* remove the original ligand atom a from $\Delta(X)$. For a point p with label $(\lambda(a), t)$, $\Phi_{at}(X, p)$ selects the n nearest protein atoms with label t. The kernel value only depends on selected points with the same label, and hence two ligand atoms a and b will be compared if and only if *(i)* they have the same atom type, and *(ii)* both their neighborhoods are constructed with respect to the protein atoms with the same atom type. This process is shown in Fig. 4(b). We denote this kernel with 3DNK_{at}.

3.3 Implementation Optimizations

A trivial implementation of this kernel would give a complexity of

$$\mathcal{O}\left(n^2 \times |\Delta(X)| \times |\Delta(Y)| \times |X| \times |Y|\right).$$

As the neighborhood function Φ in Definition 1 only depends on a single example, calculating the feature vectors can be done as a preprocessing step. Furthermore, we can optimize both preprocessing and kernel value computation:

– **Preprocessing** Our different versions of the neighborhood function Φ proposed in Sect. 3.2 depend on finding one or more nearest atoms, for which we use a k-d tree [11]. Constructing a k-d tree requires time $|X| \log |X|$, and finding n nearest neighbors requires time $n \log |X|$ for each atom. The total runtime for the preprocessing step can thus be upper bounded by

$$\mathcal{O}\left((|X| + n|S|) \log |X|\right),$$

subquadratic for values of $n < |X| / \log |X|$.

– **Kernel value approximation** Computing the actual kernel value takes quadratic time in $|S|$. However, for a point from $\Delta(X)$, only those points from $\Delta(Y)$ with a similar feature vector will significantly influence the feature vector, i.e., their representations in feature space are near. For a point a mapped to $\boldsymbol{v_a} \in \mathbb{R}^n$, we consider only those points $\boldsymbol{v_b}$ that lie within a hypercube with side $2r$ around $\boldsymbol{v_a}$. This limits the number of points but induces an error ϵ in the kernel value. The value for r for a given ϵ is then

$$r = \sigma \sqrt{2 \log \left(\frac{|S|}{\epsilon} \right)}.$$

Finding those points inside the hypercube can be done efficiently by using orthogonal range trees [17], which have a worst-case complexity of

$$\mathcal{O}\left(n|S|^{1-\frac{1}{n}} + k \right),$$

where k is number of points inside the hypercube. Note that k grows to $|S|$ as ϵ tends to zero. In our implementation we use a k-d tree as defined in [4].

The total complexity of both preprocessing and computing the kernel value can be upper bounded by

$$\mathcal{O}\left(n \times |X| \times \left(|X|^{1-\frac{1}{n}} + k \right) \right),$$

which is at most quadratic in $|X|$.

The implementation of the 3DNK kernel can be downloaded at `https://dtai.cs.kuleuven.be/software/3DNK`.

4 Experimental Evaluation

In this section, we perform an experimental analysis to assess the predictive performance of the different instantiations of 3DNK on the one hand, and w.r.t. the state-of-the-art methods on the other hand. In order to do so, we conduct experiments on four benchmark datasets (defining 31 binary classification problems and 1 regression problem) for four state-of-the-art methods.

4.1 Datasets

We assembled a first dataset (HIV) on our own, adopted two dataset collections of protein 3D structures (EC and GO) from Qiu et al. [20], and adopted a fourth dataset of 3D structures of protein-ligand pairs (PDBbind) from Ballester and Mitchell [1].

HIV Resistance (HIV) This dataset contains 2048 protein structures belonging to HIV protease. The protein sequences were retrieved from the Stanford database (`http://hivdb.stanford.edu`). 1024 sequences from patients treated with the protease inhibitor indinavir were labeled as therapy-resistant while 1024 other sequences from patients who did not receive any treatment were labeled as therapy-naive. Since there were no 3D structures available, we generated each protein's 3D structure through homology modelling, using the tool MODELLER [23] and applying standard parameters.

Enzyme Classification (EC) The Enzyme Classification (EC) dataset [10] contains 998 protein structures derived from the EC hierarchy, of which the top level consists of six enzyme classes. The benchmark contains 498 PDB structures representing these classes, plus an additional 498 PDB structures of non-enzymes. The dataset defines 7 different binary classification tasks: one predicts whether a protein structure is an enzyme, and the six others predict the correct enzyme class within the set of enzymes, adopting a *one-vs-all* strategy. The average number of examples per dataset is 569.

Gene Ontology Classification (GO) The Gene Ontology (GO) dataset links 1024 proteins to 23 GO terms [20]. All GO terms are leaves of the GO hierarchy, while 11 of them belong to the *molecular function* branch, 8 to the *biological process* branch and 4 to the *cellular component* branch. The authors transform this multi-label problem into a set of binary classification problems in the following way. For each GO term T, they partition the set of proteins into three sets. First, all proteins that are annotated with T are labeled as positive. Next, all paths from T to the root of the GO hierarchy are traversed. Any protein having a child of the terms belonging to these paths is not taken into consideration, since the authors argue that such proteins might not be properly assigned. Finally, a randomized sample of proteins (ensuring a ratio of negatives to positives of 3 to 1) of that are not on the path from T to the root are labeled as negative. The average number of examples per dataset is 173.

Protein-Ligand Interactions (PDBbind) The PDBbind benchmark dataset was designed to assess the performance of scoring functions for molecular docking. The aim is to predict whether a small molecule (called ligand) will bind to a target protein. The strength of the binding is expressed as a numerical value representing the log-value of the measured binding affinity, constituting a regression problem. Here we use the 2007 version of the PDBbind database [29], which was divided by [1] into a training set of 1105 examples and a carefully selected test set of 195 examples, which has an equal number of representatives for each protein family.

4.2 State-of-the-art Methods

We compare our method against four state-of-the-art methods that take as input graphs or 3D structures of proteins, ligands or combinations thereof. The first two are graph kernels which have been applied to biological data before, while the last two methods were designed specifically to solve the tasks of predicting protein function and protein-ligand interaction.

Fast Subtree Kernel The Fast Subtree Kernel (FSTK) is a graph kernel proposed by Shervashidze and Borgwardt [26] and is based on the Weisfeiler-Lehman test for graph isomorphism. FSTK iteratively looks at neighborhoods of nodes and unfolds the structure to get a tree-like pattern called a subtree. It then counts the matching subtree patterns of height up to h in two graphs G and G'. The authors show in their paper that FSTK outperforms four state-of-the-art graph kernels.

Fast Neighborhood Subgraph Pairwise Distance Kernel Costa and De Grave [6] propose a fast graph kernel (NSPDK) based on the pairwise distance of neighborhood subgraphs and show that it outperforms four state-of-the-art graph kernels. Their decomposition kernel works as follows. First, pairs of so-called neighborhood subgraphs are generated, and then the kernel counts the number of identical pairs of neighboring graphs of radius r at distance d between two graphs.

Mammoth Kernel Qiu et al. [20] propose a kernel that is based on the structural alignment between two proteins. However, this alignment score cannot be converted into a kernel function directly, because it is not positive-semidefinite. Instead, the authors employ an empirical kernel map as follows. For a given dataset of structures $X = x_1, \ldots, x_n$, a structure x_i is represented as an n-dimensional vector in which the jth entry is the Mammoth score between x_i and x_j. The resulting Mammoth kernel incorporates information about the alignability of a given pair of proteins. In their paper, the authors compare the Mammoth kernel to five other state-of-the-art kernels for protein structures and show that it outperforms them.

RF-Score Ballester and Mitchell [1] introduce RF-Score as an alternative to traditional scoring functions for molecular docking. RF-Score uses random forests to make predictions based on 36 features they extract from the protein-ligand pairs. Each feature is an occurrence count of a particular atom type pair (one from the ligand and one from the protein) at a maximum distance of 12 Angström from each other. In their paper, the authors show that RF-Score outperforms 18 scoring functions on a testset of 195 examples. Since we use exactly the same training and test set, we adopt the results of the different scoring functions.

4.3 Methodology

To evaluate the kernels, we generate the kernel matrices, train support vector machines (SVMs) on them and evaluate their predictive performance. As SVM implementation we use SVM^{light} [14]. To evaluate the methods on the PDBbind dataset, we compare with published results [1].

Parameter Settings We optimized the following parameters of the different methods on a separate tuning set. We tuned the regularization parameter c of the SVM out of the values $\{10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{-1}, 10^{2}, 10^{3}\}$. For 3DNK, the neighborhood size (parameter n) was tuned out of the set $\{15, 21, 27, 33\}$, and the parameter σ out of the set $\{10^{-2}, 10^{-7/4}, 10^{-3/2}, 10^{-5/4}, 10^{-1}, 10^{-3/4}, 10^{-1/2}$,- $10^{-1/4}, 10^{0}, 10^{1/4}, 10^{1/2}, 10^{3/4}, 10^{1}\}$, while the precision parameter ϵ was set to 10^{-3}. Since FSTK and NSPDK are graph kernels, we could not give them the 3D data directly. Instead, we adopted the strategy of Borgwardt et al. [3] and constructed for each protein structure a graph in which every amino acid is a node. Next, we added an edge between two amino acids if the amino acids are less than a certain distance from each other. For the protein datasets (HIV, EC and GO), we tuned FSTK and NSPDK for thresholds of 6 and 8 Angström (values suggested by the authors). For the PDBbind dataset, there are no amino acids on the ligand side, so we generated graphs with atoms as nodes instead. Since these are graphs of much smaller granularity, we added a distance threshold of 4 Angström to create graphs. For FSTK, we tuned one additional parameter h (the number of iterations) out of $\{1, ..., 11\}$. For NSPDK, we tuned two additional parameters: the distance parameter out of $\{1, 2, 3, 4\}$ and the radius parameter out of $\{0, 1, 2\}$, as recommended by the authors. For the MAMMOTH kernel, no parameters had to be tuned.

Evaluation To evaluate the classification models, we use the area under the ROC curve (AUROC) score [19]. To evaluate the regression models, we use Pearson's correlation coefficient (R), Spearman's correlation coefficient (R_S) and standard deviation of the difference between predicted and measured binding affinity (SD) in order to be able to compare with the published results of [1]. For HIV, EC and GO, a stratified 10-fold cross-validation is used. To optimize the above mentioned parameters, we constructed a tuning set through an internal 5-fold cross-validation over the training data. For PDBbind, we used the same training and test split as in [1]. Here, the parameters were optimized through a 10-fold cross-validation over the training data.

We compute the statistical significance of the different methods by computing standard deviations on the AUROC and regression scores. Method A then significantly outperforms method B at the 1% level under a t-test if their confidence intervals do not overlap.

4.4 Results

Predicting Protein Function In Table 1 we compare 3DNK$_{sw}$ and 3DNK$_{nn}$ to the state-of-the-art methods. Since RF-Score only works for protein-ligand interaction data, we could not run it for this task. First, the results show that there

Table 1. AUROC of 3DNK and the state-of-the-art methods for the benchmark classification datasets. The best scoring method per dataset is indicated in bold. For EC and GO, averaged results are reported.

Dataset	$3DNK_{sw}$	$3DNK_{nn}$	FSTK	NSPDK	Mammoth
HIV	0.848 ± 0.008	0.853 ± 0.008	0.717 ± 0.010	**0.896 ± 0.007**	0.863 ± 0.008
EC	0.575 ± 0.021	**0.600 ± 0.021**	0.573 ± 0.021	0.535 ± 0.021	0.536 ± 0.021
GO	0.744 ± 0.033	0.710 ± 0.035	0.687 ± 0.035	0.660 ± 0.036	**0.859 ± 0.026**

is no significant difference between $3DNK_{sw}$ and $3DNK_{nn}$, since their confidence intervals at 1% around their AUROC scores overlap on the three datasets. Second, $3DNK_{sw}$ and $3DNK_{nn}$ perform significantly better than FSTK on HIV, but there are no significant differences with FSTK on EC and GO. Third, $3DNK_{sw}$ and $3DNK_{nn}$ perform significantly worse than NSPDK on HIV, while $3DNK_{nn}$ significantly outperforms NSPDK on EC. On GO, there are no significant differences between NSPDK and $3DNK_{sw}$/$3DNK_{nn}$. Fourth, Mammoth significantly outperforms $3DNK_{sw}$ and $3DNK_{nn}$ on the GO dataset, while $3DNK_{nn}$ significantly outperforms Mammoth on EC. On HIV, there are no significant differences between Mammoth and $3DNK_{sw}$/$3DNK_{nn}$. The Friedman test combined with the Nemenyi post-hoc test, a non-parametric test procedure for statistical comparisons of classifiers over multiple datasets [9], also confirms that there are no significant differences between the different methods.

Predicting Protein-Ligand Interaction In Table 2 we compare $3DNK_{nn}$ and $3DNK_{at}$ to the state-of-the-art scoring functions. Since Mammoth only works on proteins, we could not apply it here. Furthermore, we could not run FSTK due to excessive memory requirements. For RF-Score (the top-scoring method), the confidence interval around its performance for R at 1% is $[0.691, 0.840]$. This shows that the performance of $3DNK_{at}$ is not significantly different than the one of RF-Score.

Conclusion The different instantiations of 3DNK perform competitively when compared to the state-of-the-art methods on the two tasks.

5 Related Work

A first group of methods that learn on geometrical data can be found in the field of pattern mining. Kuramochi and Karypis [16] present a framework in which the frequent pattern mining task is upgraded to the geometrical level. Their algorithm finds frequent geometric subgraphs (with 3D coordinates) which are rotation, scaling and translation invariant. Because noise is often present in these types of data, they perform an inexact matching based on a user-defined tolerance threshold. Nowozin and Tsuda [18] approach the task, which they call frequent subgraph retrieval, from a slightly different angle: they start from a

Table 2. Pearson's correlation coefficient (R), Spearman's correlation coefficient (R_S) and standard deviation of the difference between predicted and measured binding affinity (SD) of 3DNK and the state-of-the-art methods (including the results published in [1]) for the PDBbind benchmark regression dataset. FSTK and RF-Score could not be applied on this dataset. Methods are ordered by decreasing R.

	Method	R	R_S	SD
1	RF-Score	0.776	0.762	1.58
2	$3DNK_{at}$	0.730	0.75	1.67
3	NSPDK	0.685	0.679	1.83
4	$3DNK_{nn}$	0.652	0.688	1.81
5	X-Score::HMScore	0.644	0.705	1.83
6	$DrugScore^{CSD}$	0.569	0.627	1.96
7	SYBYL::ChemScore	0.555	0.585	1.98
8	DS::PLP1	0.545	0.588	2.00
9	GOLD::ASP	0.534	0.577	2.02
10	SYBYL::G-Score	0.492	0.536	2.08
11	DS::LUDI3	0.487	0.478	2.09
12	DS::LigScore2	0.464	0.507	2.12
13	GlideScore-XP	0.457	0.435	2.14
14	DS::PMF	0.445	0.448	2.14
15	GOLD::ChemScore	0.441	0.452	2.15
16	SYBYL::D-Score	0.392	0.447	2.19
17	DS::Jain	0.316	0.346	2.24
18	GOLD::GoldScore	0.295	0.322	2.29
19	SYBYL::PMF-Score	0.268	0.273	2.29
20	SYBYL::F-Score	0.216	0.243	2.35
21	FSTK	–	–	–
22	MAMMOTH	–	–	–

database and a query graph and look for all subgraphs of this query graph in the database, given a geometric tolerance factor.

In the context of inductive logic programming, Srinivasan et al. [27] use a logical description of the 3D coordinates and chemical properties of molecules in order to learn structure-activity relationships. Borgwardt et al. [3] introduce graph kernels for proteins. They convert protein structures into a graph, with nodes representing secondary structure elements (integrated in the nodes are chemical properties) and propose a kernel based on random walks which uses appropriate kernels on the node level to take into account their continuous attributes. The authors also use a hyperkernel to select the best kernels and their weights for a specific dataset. Shervashidze and Borgwardt [26] describe a way to convert graphs and propose an efficient graph kernel on them. Costa and De Grave [6] propose a fast graph kernel based on the pairwise distance of neighborhood subgraphs and show that it outperforms four state-of-the-art graph kernels, including the one of [26]. Ceroni et al. [5] incorporate the 3D structure directly in their decomposition kernel, but is limited to small molecules. Hinselmann et al. [12] present a graph decomposition kernel for small molecules which also takes into account 3D information. The idea is to assign each atom the distance information to the remaining atoms and their corresponding atom type. This information is stored in a trie, which holds information on the shortest path and the geometrical environment. This leads to efficient computation of the local kernels.

Qiu et al. [20] proposed the MAMMOTH kernel. In [13], the authors convert their kernel into a paired variant in order to decide whether two proteins interact with each other. Ballester and Mitchell [1] proposed RF-Score. These methods were discussed in Sect. 4.2.

6 Conclusions and Further Work

In this paper, we introduced the 3DNK kernel, which acts on 3D structures. We applied 3DNK to two biological tasks and compared it to four state-of-the-art methods. Empirical evaluation showed that 3DNK performed competitively w.r.t. the state-of-the-art graph kernels and two methods that were designed specifically to solve the two respective biological tasks. Contrary to these application-specific methods, 3DNK is more broadly applicable and can solve both tasks equally well as those methods. The results suggest that the information in 3D structures can be exploited successfully and that the kernel can be deployed on a variety of problems.

In future work, we will explore various aspects of the 3DNK family further (such as the parameter space) and search for application domains on which new instantiations can be applied.

Acknowledgements

The authors would like to thank students Davy De Mits and Sunil Aryal for conducting preliminary experiments, Dr. Kurt De Grave and Dr. Fabrizio Costa for assistance with running NSPDK, and Jérôme Renaux for proofreading. This research was supported by ERC-StG 240186 MiGraNT and IWT-SBO Nemoa.

References

1. P.J. Ballester, J.B.O. Mitchell. A machine learning approach to predicting protein-ligand binding affinity with applications to molecular docking. *Bioinformatics*, 26(9):1169–1175, 2010.
2. K. Borgwardt. *Graph Kernels.* PhD thesis, Computer Science, Ludwig-Maximilians-University Munich, 2007.
3. K. Borgwardt, C. Ong, S. Schonauer, S. Vishwanathan, A. Smola, and H. Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(S1):i47–i56, 2005.
4. M. de Berg, O. Cheong, M. Kreveld, and M. Overmars Computational Geometry: Algorithms and Applications *Springer-Verlag*, 2000.
5. A. Ceroni, F. Costa, and P. Frasconi. Classification of small molecules by two- and three-dimensional decomposition kernels. *Bioinformatics*, 23(16):2038–2045, 2007.
6. F. Costa and K. De Grave. Fast neighborhood subgraph pairwise distance kernel. In *Proceedings of the 27th International Conference on Machine Learning*, 2010.
7. N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other Kernel Based Methods.* Cambridge University Press, UK, 2000.
8. K. Deforche. *Modeling HIV resistance evolution under drug selective pressure.* PhD thesis, Katholieke Universiteit Leuven, 2008.
9. J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(Jan):1–30, 2006.
10. P.D. Dobson and A.J. Doig. Predicting enzyme class from protein structure without alignments. *J. Mol. Biol.*, 345:187–199, 2005.
11. J.H. Friedman, J.L. Bentley, and R.A. Finkel. An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Trans. Math. Softw.*, 98:209–226, 1977.

12. G. Hinselmann, N. Fechner, A. Jahn, M. Eckert, and A. Zell. Graph kernels for chemical compounds using topological and three-dimensional local atom pair environments. *Neurocomputing*, 74:219–229, 2010.
13. M. Hue, M. Riffle, J.-P. Vert, and W. Stafford Noble. Large-scale prediction of protein-protein interactions from structures. *BMC Bioinformatics*, 11(144), 2010.
14. T. Joachims. *Learning to Classify Text using Support Vector Machines: Methods, Theory, and Algorithms*. Springer, 2002.
15. R.D. King, S. Muggleton, A. Srinivasan, and M.J.E. Sternberg. Structure-activity relationships derived by machine learning: The use of atoms and their bond connectivities to predict mutagenicity by inductive logic programming. *Proceedings of the National Academy of Sciences*, 93:438–442, 1996.
16. M. Kuramochi and G. Karypis. Discovering frequent geometric subgraphs. In *Proceedings of the 2004 IEEE International Conference on Data Mining*, pages 258–265, 2004.
17. D.T. Lee and C.K. Wong Worst-case analysis for region and partial region searches in multidimensional binary search trees and balanced quad trees. *Acta Informatica*, 9:23–29, 1977.
18. S. Nowozin and K. Tsuda. Frequent subgraph retrieval in geometric graph databases. In *Proceedings of the 2008 IEEE International Conference on Data Mining*, pages 953–958, 2008.
19. F. Provost and T. Fawcett. Analysis and visualization of classifier performance: comparison under imprecise class and cost distributions. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 43–48. AAAI Press, 1998.
20. J. Qiu, M. Hue, A. Ben-Hur, J.-P. Vert, and W. Stafford Noble. A structural alignment kernel for protein structures. *Bioinformatics*, 23(9):1090–1098, 2007.
21. J. Ramon and T. Gärtner. Expressivity versus efficiency of graph kernels. In *Proceedings of the first international workshop on mining graphs, trees and sequences (MGTS2003)*, pages 65–74, 2003.
22. R. Saidi, M. Maddouri, and E.M. Nguifo. Comparing graph-based representations of protein for mining purposes. In *Proceedings of the KDD-09 Workshop on Statistical and Relational Learning in Bioinformatics*, pages 35–38, 2009.
23. A. Sali and T.L. Blundell. Comparative protein modelling by satisfaction of spatial restraints. *Journal of Molecular Biology*, 234:779–815, 1993.
24. L. Schietgat, C. Vens, J. Struyf, H. Blockeel, D. Kocev, and S. Džeroski. Predicting gene function using hierarchical multi-label decision tree ensembles. *BMC Bioinformatics*, 11(2), 2010.
25. L. Schietgat, J. Ramon and M. Bruynooghe. A polynomial-time maximum common subgraph algorithm for outerplanar graphs and its application to chemoinformatics. *Annals of Mathematics and Artificial Intelligence*, 69:343–376, 2013.
26. N. Shervashidze and K. Borgwardt. Fast subtree kernels on graphs. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1660–1668, 2009.
27. A. Srinivasan, D. Page, R. Camacho, and R.D. King. Quantitative pharmacophore models with inductive logic programming. *Machine Learning*, 64:65–90, 2006.
28. J. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific, Singapore, 2005.
29. R. Wang et al. The PDBbind database: methodologies and updates. *J. Med. Chem.*, 48:4111-4119, 2005.